# Two-Level Path Authentication in EPCglobal Network

Hongbing Wang
Shanghai Jiao Tong University;
Singapore Management University
Email: flora_wang@sjtu.edu.cn

Yingjiu Li
Singapore Management University
Email: yjli@smu.edu.sg

Zongyang Zhang, Zhenfu Cao
Shanghai Jiao Tong University
Email: zongyangzhang@sjtu.edu.cn,
zfcao@cs.sjtu.edu.cn

***Abstract*—In this paper, we propose a two-level path authentication protocol for object genuineness verification in RFID-based supply chain and EPCglobal Network. In our solution, a tag's path in a supply chain can be generated dynamically, where each reader in the path can verify the validation of the path using its own private key. Our solution has a few promising properties, including dynamic path generation, distributed authentication, and scalability. In comparison, the previous path authentication solution for RFID tags is focused on static path generation and centralized control. The efficiency of our path authentication protocol is enhanced significantly by dividing a whole path into multiple segments according to organization structures. The security and privacy of our protocol are established based on two cryptographic primitives, hierarchical identity-based encryption and batch verification signature.**

***Index Terms*—RFID, path authentication, EPCglobal Network**

## I. Introduction

The EPCglobal Network is a computer network which enables organizations to capture, share, and discover product data, including Electronic Product Codes (EPC) and related business events using EPCglobal certified hardware and software components and standard interfaces. The EPCglobal Network facilitates true visibility of information about items in supply chains, thus making organizations more effective. In the EPCglobal Network, the EPC serves as a globally unique identifier that identifies product items moving between suppliers, distributors, shippers and end users at two levels: either within organizations, where EPC related information is stored in EPC Information Services (EPCIS), or between organizations, where EPC related information in EPCIS can be queried via EPC discovery Services (EPCDS). Radio Frequency Identification (RFID) technology is widely used in EPCglobal Network, where EPC codes are stored in RFID tags, collected by RFID readers, and transmitted with associated business event data through RFID networks [1].

Among many applications [2], RFID-enabled supply chain management is one of the major applications in the EPCglobal Network. In RFID-enabled supply chain management, path authentication is especially important, for it helps protect object genuineness by maintaining object pedigree and the supply chain integrity [3]. It can be used to thwart counterfeiting, cloning, and replica of luxury products or pharmaceutics, which are major concerns of supply chain parties [4], [5], [6], [7], [8], [3]. Effective path authentication can be used to verify genuine products according to the valid paths by which they have been processed in RFID-enabled supply chains.

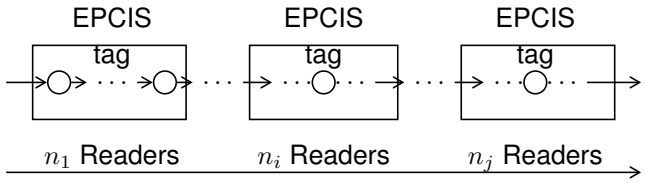


Fig. 1. Division of path authentication into segments

In NDSS'11, Blass et al. [3] proposed the first solution to path authentication in an RFID-based supply chain. The solution is both secure, which means it is verifiable whether an object or tag has taken a valid path, and privacy preserving, which means it is impossible for an adversary to identify or trace any legitimate tag in operation. While this solution started an important direction in the research on security and privacy in RFID-enabled supply chain [9], [10], [11], [12], [13], [14], [2], [3], it has certain limitations, including: (1) *Centralized control*: there is only one last point, called manager or checkpoint, which can verify the validation of a tag and its path; (2) *Single point of failure*: the issuer of a supply chain who prepares all tags for deployment generates secret keys for all readers as well as for the manager in a valid path. All these keys are sent to the manager by the issuer; therefore, the manager represents a single point of failure; and (3) *Static path configuration*: the issuer sends private keys of readers as well as their positions in a valid path to the manager, which means that the path is determined in advance. These limitations heavily restrict the application of this solution. For example, a valid path in real-world applications may not be fixed due to many reasons such as reader damage and dysfunction of certain conveyance point. It is thus critical to devise a path authentication solution which does not have the limitations. This was also left as an open and challenging problem in [3]. We mainly engage in resolving this problem in this paper.

In this paper, we present a path authentication protocol for RFID-based supply chain in EPCglobal Network. In our solution, *every node* in a path can verify the validation of the path and the path can be generated dynamically. There is no centralized control, and thus no single point of failure in our

| | **NDSS'11** [3] | **Our Solution** |
|---|---|---|
| Path generation | static | dynamic |
| Path authentication | centralized | distributed |
| Single point of failure | yes | no |
| Scalability | no | yes |
| Secrets keys | checkpoint with all | each node with its own |
| Tag memory | 960 bits | $960 + 4 * 160$ bits |

TABLE I
COMPARISON BETWEEN BLASS ET AL.'S SOLUTION AND OURS

solution. We divide a long valid path into several segments according to organization structures. Each organization is represented by an EPCIS in EPCglobal Network. The path authentication consists of two parts: tag path authentication within an EPCIS and path authentication among EPCIS. By using our two-level path authentication methods, the efficiency of verification of a tag's path is improved significantly. For example, suppose that a tag goes through $j$ EPCIS, and it is read by $n_1, \cdots, n_j$ readers in each EPCIS, respectively, as illustrated in Fig. 1. If the previous solution is used, the last reader needs to verify the path from the very beginning to the end. Using our two-level path authentication, however, we could divide the $n_1 + \cdots + n_j$ readers authentication into $j$ segments according to EPCIS. Each EPCIS only needs to verify its previous EPCIS's signature on the path before it authenticates the path within its own EPCIS. Moreover, each EPCIS can batch verify signatures on valid paths for tags handed over from the previous EPCIS, which further reduces the computational costs.

### A. Our Contribution

Compared to the previous work on tag path authentication [3], our major contribution is clarified as below:

- We propose the first *two-level* path authentication solution in an EPCglobal Network. Our solution is suitable for dynamic and large-scale supply chain with distributed control. In comparison, the previous work [3] is targeted at *one-level* static path authentication with centralized authentication.
- In our solution, a tag's path can be *dynamically* generated. A reader is able to choose any reader as its next reader for a designated tag, while in [3], the path is *static* and *determined in advance*. It means that in [3], a reader can only choose its next reader on the predetermined path.
- Our solution is *distributed* and more *scalable*. Each reader in a path can use its private key to verify the path of a designated tag. On the contrary, the previous work [3] is *centralized* and *non-scalable*: only the checkpoint with all secret keys could verify a tag's path.
- There is *no single point of failure* in our solution. Each reader is equipped with its own private key only. It is obvious that if a reader in a path is corrupted, then only its forward path is disclosed; however, the backward path is still secure. In [3], as the manager owns all the private keys, all paths are disclosed if the manager is corrupted.

Table I summarizes the comparison between the previous work [3] and ours. We achieve all desirable features at the cost of 640 bits more tag memory than the previous work [3].

### B. Our Technique

In the first level of our protocol, tag path authentication within EPCIS, we exploit hierarchical identity-based encryption (HIBE) with constant size ciphertext [15]. In our solution, all tag's paths in an EPCIS are constructed as an HIBE tree. All the readers are possible nodes in an HIBE tree. The issuer is denoted as the root of the HIBE tree. The issuer computes the initial state for a tag and writes the initial state to the tag in the following steps: (1) choose a reader as its son reader; (2) generate a private key for this chosen reader; (3) encrypt the tag's identity, or EPC, using its chosen reader's public key; and (4) write the encrypted information - initial state to the tag. When the above reader interacts with the tag, it reads out the current state of the tag, and re-encrypts the state to a new state under next reader's public key. By doing this continuously, the tag proceeds within the EPCIS. In our solution, each reader may have more than one private keys. However, by checking the signature which is a part of the state of a tag, a reader can find out its parent, thus find out the right path and the right private key which will be used in decrypting the state to get the tag's identity (please refer to Section III-A3).

In order to facilitate the second level of path authentication among EPCIS, we require that each leaf reader (i.e., an exit point of current EPCIS) in an HIBE tree first decrypt the state to get the identity of the tag and its path. Then, the leaf reader verifies the signature stored on the tag to determine whether the tag is tampered. If the verification succeeds, the leaf reader encrypts the tag as well as its path (as a whole message) using the current EPCIS's public key and signs on this encrypted message. Before the tag moves out of this EPCIS and enters to the region of the next EPCIS for further disposal, the current EPCIS should authenticate the tag and its path, then encrypt tag and its path using the next EPCIS's public key. Meanwhile, the current EPCIS should sign the tag and its path to ensure its integrity. The next EPCIS only needs to verify the tag and its path within the previous EPCIS by checking the previous EPCIS's signature. This reduces computational costs significantly, because it does not need to verify the tag's path for each reader within the previous EPCIS. Moreover, the path authentication among EPCIS can be performed with batch verification signature techniques. Therefore, numerous tags and their paths can be verified in batches for high computational efficiency.

**Organization.** The rest of this paper is organized as follows: In Section II, preliminary knowledge, including definitions and security models are defined. Then, our two-level path authentication protocol is presented in Section III, followed by security analysis in Section IV. Finally, Section V concludes this paper.

## II. Preliminaries and Definitions

### A. Components

Tags $\{T_i\}$: Tags are radio transponders attached to physical objects. Each tag has an initial state written by issuer, and the state is updated each time when it interacts with a reader.

Issuer $I$: There is only one issuer in each EPCIS. The issuer is responsible for the generation of system's public parameters, master security key, and the private key of its EPCIS. For a new tag $T$ which is ready to enter a supply chain, $I$ writes an initial state to $T$.

Readers $\{R_i\}$: Each reader interacts with numerous tags. It reads out the current state of a tag, and then computes a new state for the tag. At last, the new state is written to the tag. Each reader has its back-end database for computations and storage. In our solution, every reader can verify a tag's path when the tag is read by the reader.

EPCIS: EPCIS enables disparate applications to leverage EPC data via EPC-related data sharing, both within and across enterprises. In our solution, each EPCIS first verifies the validation of tags and their paths within it. Then the EPCIS encrypts and signs the valid tags and their paths for the next EPCIS to let these tags be disposed by the next EPCIS in a supply chain.

EPCDS: EPCDS is a referral system that enables a client to find multiple information resources about an object if the object can be uniquely identified with its EPC. In our solution, the public parameters of each EPCIS are published on EPCDS.

### B. Path Authentication Protocol in RFID-based Supply Chain

Our path authentication protocol consists of two levels: (1) within EPCIS and (2) among EPCIS.

*1) Within EPCIS:* This part is a revision of Boneh et al.'s HIBE scheme [15], which is described as follows:

$\mathsf{Setup}(1^n) \rightarrow (\mathsf{para}, \mathsf{msk})$: Each issuer $I$, which is the root of an HIBE tree in an EPCIS, is responsible for the system's setup. This algorithm generates system's public parameters para and master secret key msk. It also generates a private key for its EPCIS, and sends the private key to the EPCIS via a secure channel. After the generation of para, the issuer publishes the system's public parameters on EPCDS, and keeps the msk secret.

$\mathsf{KeyGen}(\mathsf{para}, d_{\mathsf{ID}_{k-1}}, \mathsf{ID}_{R_k}) \rightarrow d_{\mathsf{ID}_k}$: Given para for the system, private key $d_{\mathsf{ID}_{k-1}}$ of a parent reader, and a son's identity $\mathsf{ID}_{R_k}$, it generates the son's (reader $R_k$'s) private key $d_{\mathsf{ID}_k}$.

$\mathsf{SetSecretKey}(\mathsf{para}, R_i) \rightarrow x_{R_i}$: On input of the public parameters para for the system, it outputs $x_{R_i}$ as reader $R_i$'s secret value.

$\mathsf{SetPublicKey}(\mathsf{para}, x_{R_i}) \rightarrow \mathsf{PK}_{R_i}$: On input of para and secret value $x_{R_i}$ of reader $R_i$, it outputs public key as $\mathsf{PK}_{R_i}$ for reader $R_i$.

$\mathsf{Enc}(\mathsf{para}, \mathsf{msk}, \mathsf{ID}_{R_i}, \mathsf{ID}_T) \rightarrow C_{\mathsf{ID}_{R_i}}$: Given the system's parameters para, master secret key msk, a public key $\mathsf{ID}_{R_i}$ of reader $R_i$, and a tag $T$'s identity $\mathsf{ID}_T$ in the message space, the encryption algorithm Enc outputs a ciphertext $C_{\mathsf{ID}_{R_i}}$ of $T$ under public key $\mathsf{ID}_{R_i}$.

$\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_{R_j}) \rightarrow C_{\mathsf{ID}_{R_j}}$: Given the system's public parameters para, a secret value $x_{R_i}$ of reader $R_i$, a ciphertext under public key $\mathsf{ID}_{R_i}$ of reader $R_i$, and a public key $\mathsf{ID}_{R_j}$ of reader $R_j$, the algorithm ReEnc outputs a ciphertext $C_{\mathsf{ID}_{R_j}}$ under $\mathsf{ID}_{R_j}$.

$\mathsf{Dec}(\mathsf{para}, d_{\mathsf{ID}_i}, C_{\mathsf{ID}_{R_i}}) \rightarrow (T, \perp)$: Given para, reader $R_i$'s private key $d_{\mathsf{ID}_i}$, and a ciphertext $C_{\mathsf{ID}_{R_i}}$ under reader $R_i$'s identity $\mathsf{ID}_{R_i}$, the decryption algorithm Dec outputs the corresponding tag $T$ in the message space or the error symbol $\perp$.

*2) Among EPCISs:* This part of our solution is based on Camenisch et al.'s [16] batch verification of BLS signature [17]. However, we modify it from many signers on many messages to a single signer on many messages. It is described as follows:

$\mathsf{KeyGen}(\mathsf{para}) \rightarrow (\mathsf{svk}, \mathsf{ssk})$: Given the system's parameters para, the algorithm outputs the verification/signing key pair as $(\mathsf{svk}, \mathsf{ssk})$.

$\mathsf{Sign}(\mathsf{para}, \mathsf{ssk}, m) \rightarrow (m, \sigma)$: Given the system's parameters para, the signing secret key ssk, and a message $m$ from the message space, it outputs a signature on $m$ as $\sigma$.

$\mathsf{Verify}(\mathsf{para}, \mathsf{svk}, m_i, \sigma_i) \rightarrow (\mathsf{accept}, \mathsf{reject}), i = 1, \cdots, n$: Given para, a verification key svk, and $n$ purported signatures $\sigma_i$ from a single user on messages $m_i$ for $i = 1, \cdots, n$, the algorithm outputs accept if the verification is successful; otherwise it outputs reject.

### C. Security Statements and Adversary Model

We use game-based security model to describe the security of our protocol. First, we describe several oracles which any probabilistic polynomial-time (PPT) adversary could query during the security game.

$\mathcal{O}_{\mathsf{rd}}(T_i)$: On the query of a tag $T_i$, the challenger returns the current state of $T_i$ to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{enc}}(T_i, R_j)$: On the query of a tag $T_i$ and an arbitrary reader $R_j$ chosen by the adversary $\mathcal{A}$, the challenger responds to $\mathcal{A}$ by running $\mathsf{Enc}(\mathsf{para}, \mathsf{msk}, \mathsf{ID}_{R_j}, \mathsf{ID}_{T_i})$, where $\mathsf{ID}_{R_j}$ and $\mathsf{ID}_{T_i}$ are the identities of reader $R_j$ and tag $T_i$, respectively.

$\mathcal{O}_{\mathsf{reenc}}(C_{\mathsf{ID}_{R_i}}, R_j)$: On the query of a state $C_{\mathsf{ID}_{R_i}}$ and an arbitrary reader $R_j$ chosen by the adversary $\mathcal{A}$, the challenger responds to $\mathcal{A}$ by running $\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_{R_j})$.

$\mathcal{O}_{\mathsf{val}}(T_i)$: On the query of a path that a tag $T_i$ goes through, the challenger returns 1 to the adversary $\mathcal{A}$ if tag $T_i$ goes through a valid path; otherwise, the challenger returns 0 to the adversary $\mathcal{A}$. Note that the challenger does not return the real path to $\mathcal{A}$.

$\mathcal{O}_{T,P}(P_i)$: On input a specified valid path $P_i$ , the challenger randomly selects a tag from the path $P_i$, and returns the identity of the tag to $\mathcal{A}$.

$\mathcal{O}_{T,R}(R_i)$: On input a specific step $R_i$, the challenger randomly picks a tag which has gone through step $R_i$, and returns the identity of the tag to $\mathcal{A}$.

We consider authentication, tag privacy, and (tag and path) unlinkability for RFID applications in a supply chain in our protocol.

*1) Authentication:*

**Definition II.1** (Authentication)**.** Authentication implies that any PPT adversary cannot forge a tag's internal state with a valid path that has not actually been taken by the tag.

**Definition II.2** (Authentication Model)**.** The PPT adversary $\mathcal{A}$ is given the system's public parameters before he interacts with the challenger $\mathcal{C}$ in the following game.

**Select:** $\mathcal{A}$ selects a target step $v^*$, where $v^*$ is associated with certain reader $R^*$.

**Phase 1:** $\mathcal{A}$ adaptively makes the following queries on a tag or the tag's state:

- $\mathcal{O}_{\mathsf{enc}}(T_i,\ R_j)$: For the initial state query of tag $T_i$ and an arbitrary reader $R_j$ selected by $\mathcal{A}$, the challenger $\mathcal{C}$ returns $\mathsf{Enc}(\mathsf{para}, \mathsf{msk}, \mathsf{ID}_{R_j}, \mathsf{ID}_{T_i})$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{reenc}}(C_{\mathsf{ID}_{R_i}},\ R_j)$: For the current state $C_{\mathsf{ID}_{R_i}}$, $\mathcal{A}$ makes request to $\mathcal{C}$ for a new state of tag $T_i$ under his chosen reader $R_j$. The challenger $\mathcal{C}$ returns to $\mathcal{A}$ the output $\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_{R_j})$.
- $\mathcal{O}_{\mathsf{rd}}(T_i)$: $\mathcal{A}$ requests on the current state of tag $T_i$; the challenger $\mathcal{C}$ reads out the current state of tag $T_i$ and returns it to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{val}}(T_i)$: On input of a tag $T_i$, the challenger returns 1 to the adversary $\mathcal{A}$ if tag $T_i$ goes through a valid path; otherwise, the challenger $\mathcal{C}$ returns 0 to $\mathcal{A}$. However, the challenger does not return the real path to $\mathcal{A}$.

**Challenge:** $\mathcal{A}$ selects a tag $T_c$ as his target, and outputs a forged state of tag $T_c$ at step $r$, (i.e., reader $R_r$) as $C_{\mathsf{ID}_{R_r}}$.

**Decision:** $\mathcal{C}$ outputs 0 if $T_c$ has not gone through the step $v^* \in P$, where $P$ is the path $T_c$ took; otherwise, he outputs 1.

**Definition II.3.** Let adv denote the event that $\mathcal{A}$ outputs a valid tag state $C_{\mathsf{ID}_{R_r}}$ as he claims in the above security game. A path authentication solution is *authenticated* if for all PPT adversary $\mathcal{A}$, $\Pr[\mathsf{adv}] \leq \varepsilon$ holds, where $\varepsilon$ is negligible.

*2) Tag Privacy:*

**Definition II.4** (Tag Privacy)**.** A path authentication solution achieves tag privacy, if any PPT adversary $\mathcal{A}$ cannot distinguish whether or not a tag $T_i$ has passed some step $R$ based on the data stored on the tag only [3].

**Definition II.5** (Privacy Game)**.** The PPT adversary $\mathcal{A}$ is given the system's public parameters before he interacts with challenger $\mathcal{C}$ in the following game.

**Select:** $\mathcal{A}$ chooses an arbitrary reader $R^*$ as his target.

**Phase 1:** In this phase, $\mathcal{A}$ makes the following queries to $\mathcal{C}$:

Let $S$ be an initially empty set recording the tags which $\mathcal{A}$ has queried during the game.

- $\mathcal{O}_{\mathsf{enc}}(T_i, R_j)$: $\mathcal{A}$ chooses an arbitrary tag $T_i$ and a reader $R_j$ ($R_j$ may or may not be $R^*$), and queries on the initial state of tag $T_i$ under reader $R_j$; the challenger $\mathcal{C}$ returns $C_{\mathsf{ID}_{R_j}} \leftarrow \mathsf{Enc}(\mathsf{para}, \mathsf{msk}, \mathsf{ID}_{R_j}, \mathsf{ID}_{T_i})$. If $R_j = R^*$, $\mathcal{C}$ sets $S \leftarrow S \bigcup \{T_i\}$.
- $\mathcal{O}_{\mathsf{reenc}}(C_{\mathsf{ID}_{R_i}}, R_j)$: For a tag's state under reader $R_i$, $\mathcal{A}$ makes request to $\mathcal{C}$ for a new state under reader $R_j$. $\mathcal{C}$ returns $\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_{R_j})$ to $\mathcal{A}$. If $R_i = R^*$, or $R_j = R^*$, $\mathcal{C}$ computes $T_i \leftarrow \mathsf{Dec}(\mathsf{para}, d_{\mathsf{ID}_i}, C_{\mathsf{ID}_{R_i}})$, and sets $S \leftarrow S \bigcup \{T_i\}$.
- $\mathcal{O}_{\mathsf{rd}}(T_i)$: $\mathcal{A}$ requests on the current state of tag $T_i$; $\mathcal{C}$ reads out the current state of tag $T_i$ and returns it to $\mathcal{A}$. If tag $T_i$ is under reader $R^*$, $\mathcal{C}$ sets $S \leftarrow S \bigcup \{T_i\}$.
- $\mathcal{O}_{\mathsf{val}}(T_i)$: On input of a tag $T_i$, the challenger returns 1 to the adversary $\mathcal{A}$ if tag $T_i$ goes through a valid path; otherwise, the challenger $\mathcal{C}$ returns 0 to $\mathcal{A}$. Note that the challenger does not return the real path to $\mathcal{A}$.
- $\mathcal{O}_{T,R}(R_i)$: On input a specific reader $R_i$, the challenger randomly picks a tag $T_j$ which has gone through step $R_i$, and returns it to $\mathcal{A}$. If $R_i = R^*$, $\mathcal{C}$ sets $S \leftarrow S \cup \{T_j\}$.

**Challenge:** $\mathcal{C}$ chooses a random bit $b$ from $\{0, 1\}$. If $b$ = 0, $\mathcal{C}$ selects a tag $T_c$ which has not passed through $R^*$. Otherwise, if $b$ = 1, $\mathcal{C}$ randomly selects a tag $T_c$ which went through $R^*$. Then, $\mathcal{C}$ reads out the current state of $T_c$, and sends it to $\mathcal{A}$ as his challenge. The restriction here is that $T_c \notin S$.

Decision: $\mathcal{A}$ outputs his guess $b' = 1$ if $T_c$ has passed through $R^*$. Otherwise, he outputs $b' = 0$.

**Definition II.6.** Let adv denote the event that $\mathcal{A}$ outputs a right guess in the decision phase. A path authentication solution is *privacy preserving*, if for any PPT adversary $\mathcal{A}$, $\Pr[\mathsf{adv}] \leq \frac{1}{2} + \varepsilon$ holds, where $\varepsilon$ is negligible.

*3) Unlinkability:* In accordance with [3], unlinkability can be defined in terms of tag unlinkability and path unlinkability.

*a) Tag unlinkability:* Tag unlinkability means that given some states of two arbitrary tags $T_0$ and $T_1$ in a supply chain, no PPT adversary $\mathcal{A}$ can distinguish $T_0$ from $T_1$ with non-negligible event.

**Definition II.7** (Tag Unlinkability Game)**.** A PPT adversary $\mathcal{A}$ is given the system's public parameters before he interacts with the challenger $\mathcal{C}$ in the following game.

**Select:** The adversary $\mathcal{A}$ chooses two random tags $T_0$ and $T_1$.

**Phase 1:** $\mathcal{A}$ makes the following queries:

- $\mathcal{O}_{\mathsf{enc}}(T_i, R_j)$: $\mathcal{A}$ chooses an arbitrary reader $R_j$, and queries on the initial state of tag $T_i$ under reader $R_j$. The challenger $\mathcal{C}$ returns $C_{\mathsf{ID}_{R_j}} \leftarrow \mathsf{Enc}(\mathsf{para}, \mathsf{msk}, \mathsf{ID}_{R_j}, \mathsf{ID}_{T_i})$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{reenc}}(C_{\mathsf{ID}_{R_i}}, R_j)$: On input tag $T_i$'s state under reader $R_i$, $\mathcal{A}$ makes request to $\mathcal{C}$ for a new state under reader $R_j$. $\mathcal{C}$ returns $\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_{R_j})$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{rd}}(T_i)$: $\mathcal{A}$ requests on the current state of tag $T_i$; $\mathcal{C}$ reads out the current state of tag $T_i$ and returns it to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{val}}(T_i)$: On input of a tag $T_i$, $\mathcal{C}$ returns 1 to $\mathcal{A}$ if tag $T_i$ goes through a valid path; otherwise, $\mathcal{C}$ returns 0 to $\mathcal{A}$. Note that $\mathcal{C}$ does not return the real path to $\mathcal{A}$.
- $\mathcal{O}_{T,P}(P_i)$: On input a specific path $P_i$ which both $T_0$ and $T_1$ has not gone through, $\mathcal{C}$ randomly picks a tag which has passed through the path $P_i$, and returns it to $\mathcal{A}$.

**Challenge:** First, $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$. Second, $\mathcal{C}$ reads out the the current state $C_{\mathsf{ID}_{R_i}}$ of $T_b$. Finally, $\mathcal{C}$ updates $C_{\mathsf{ID}_{R_i}}$ to $C_{\mathsf{ID}_{R_j}}$ using $\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_{R_j})$, and sends $C_{\mathsf{ID}_{R_j}}$ to $\mathcal{A}$ as the challenge.

Decision: $\mathcal{A}$ outputs his guess $b'$. If $b' = b$, $\mathcal{A}$ is said to be successful in the game. Otherwise, $\mathcal{A}$ fails.

**Definition II.8.** Suppose that adv defines the event that $\mathcal{A}$ outputs a right guess in the decision phase. A path authentication solution is *tag unlinkable*, if for any PPT adversary $\mathcal{A}$, $\Pr[\mathsf{adv}] \leq \frac{1}{2} + \varepsilon$ holds, where $\varepsilon$ is negligible.

*b) Path unlinkability.:* It means that given two tags $T_i$ and $T_j$, every PPT adversary $\mathcal{A}$ can tell whether these two tags go through the same path with probability at most $\frac{1}{2} + \varepsilon$, where $\varepsilon$ is negligible.

**Definition II.9** (Path Unlinkability Game)**.** The PPT adversary $\mathcal{A}$ is given the system's public parameters before he interacts with the challenger $\mathcal{C}$ in the following game.

**Select:** $\mathcal{A}$ chooses a random tag $T$. $\mathcal{C}$ then gives the path $P$ which $T$ goes through, as well as each state of $T$ in that path to $\mathcal{A}$.

**Phase 1:** $\mathcal{A}$ makes the following queries:

- $\mathcal{O}_{\mathsf{enc}}(T_k,\ R_j)$: $\mathcal{A}$ chooses an arbitrary reader $R_j$, and queries on the initial state of tag $T_k$ under reader $R_j$. $\mathcal{C}$ returns $C_{\mathsf{ID}_{R_j}} \leftarrow \mathsf{Enc}(\mathsf{para}, \mathsf{msk}, \mathsf{ID}_{R_j}, \mathsf{ID}_{T_k})$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{reenc}}(C_{\mathsf{ID}_{R_i}},\ R_j)$: On input tag $T_k$'s state under reader $R_i$, $\mathcal{A}$ makes request to $\mathcal{C}$ for a new state of $T_k$ under reader $R_j$. $\mathcal{C}$ returns $\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_{R_j})$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{rd}}(T_i)$: $\mathcal{A}$ requests on the current state of tag $T_i$; $\mathcal{C}$ reads out the current state of tag $T_i$ and returns it to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{val}}(T_i)$: On input of a tag $T_i$, $\mathcal{C}$ returns 1 to the adversary $\mathcal{A}$ if tag $T_i$ goes through a valid path; otherwise, $\mathcal{C}$ returns 0 to $\mathcal{A}$. Note that $\mathcal{C}$ does not return the real path to $\mathcal{A}$.
- $\mathcal{O}_{T,P}(P_i)$: On input of a valid specific path $P_i \neq P$ by the adversary, the challenger randomly picks a tag which goes through the path $P_i$, and returns the tag to $\mathcal{A}$.

**Challenge:** First, $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$ and a challenge tag $T_c$. If $b = 0$, $\mathcal{C}$ randomly selects a tag $T_c$ which does not go through $P$. Otherwise, if $b = 1$, $\mathcal{C}$ random selects a tag $T_c \neq T$ which goes through $P$. Second, $\mathcal{C}$ reads out the current state of $T_c$, i.e., $C_{\mathsf{ID}_{R_i}}$, and updates $C_{\mathsf{ID}_{R_i}}$ into a new state $C_{\mathsf{ID}_{R_j}}$ which is under an arbitrary reader $R_j$ selected by the challenger $\mathcal{C}$.

**Decision:** $\mathcal{A}$ outputs his guess $b'$ which indicates whether $T_c$ goes through $P$. If $b' = 1$, $\mathcal{A}$ guesses $T_c$ goes through $P$. Otherwise, $\mathcal{A}$ guesses $T_c$ does not go through $P$.

**Definition II.10.** Suppose that adv defines the event that $\mathcal{A}$ outputs a right guess in the decision phase. A path authentication solution is *path unlinkable*, if for any PPT adversary $\mathcal{A}$, $\Pr[\mathsf{adv}] \leq \frac{1}{2} + \varepsilon$ holds, where $\varepsilon$ is negligible.

## III. Our Solution

In our solution, we assume that all readers are trusted and independent, i.e., each reader runs the protocol honestly, and is independent of other readers.

Our solution includes two-level path authentication, tag path authentication and EPCIS path authentication, which are illustrated in Fig. 2.

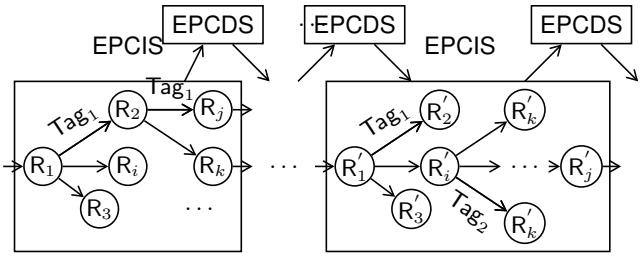


Fig. 2. Two-level path authentication architecture

### A. Tag Path Authentication within EPCIS

We make the following assumptions in our system:

1) An issuer (e.g., a manufacturer) is denoted as the root in a tree.
2) A route in a tree represents a path from the issuer to a reader within an organization.
3) Since a tag's proceeding steps within an organization (EPCIS) are usually definite, the depth of the HIBE tree is definite, too. (For flexibility, we can set the depth a little bigger which results in the system's public parameters a little longer.) According to [15], the identities of the nodes (readers in our scenario), which act as public keys, are marked as vectors of elements in $(\mathbb{Z}_p^*)^k$, written as $\mathsf{ID} = (I_1, \cdots, I_k) \in (\mathbb{Z}_p^*)^k$, where $k$ is the depth of the nodes in the tree. The first $k-1$ elements are identical to the elments of a node's parent. This implies that the identity of reader $R$ in depth one is a vector of one element $I_1$ in $\mathbb{Z}_p^*$. If $R$ appears in depth two, then its identity is marked as vector of two elements $(I_1', I_1)$ in $(\mathbb{Z}_p^*)^2$, where $I_1'$ is $R$'s parent's identity. If reader $R$ appears in depth $i$, its identity is marked as vector of $i$ elements $(I_1', I_2', \cdots, I_{i-1}', I_1)$ in $(\mathbb{Z}_p^*)^i$, where $(I_1', I_2', \cdots, I_{i-1}')$ is the identity of $R$'s parent.

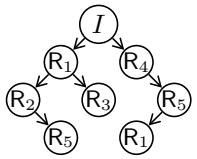


Fig. 3. Structure of HIBE tree

In Fig. 3, the alphabet $I$ represents the issuer, and the alphabet $R_i$, ($i$ = 1, $\cdots$ 5), represents a reader in EPCIS. We construct our tag path authentication based on the HIBE scheme [15].

*1) Description of HIBE Scheme:* Let $\mathbb{G}$ be a bilinear group of prime order $p$ and let $e$: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map. For security purpose, we need $|\mathbb{G}| \geq 160$ bits and $|\mathbb{G}_1| \geq 960$ bits. Furthermore, we assume that the identities in depth $k$ of an HIBE tree are vectors of elements in $(\mathbb{Z}_p^*)^k$, denoted by $\mathsf{ID} = (I_1, \cdots, I_k) \in (\mathbb{Z}_p^*)^k$, where $(I_1, \cdots, I_{k-1})$ is identity of ID's parent. For an element $I_i$ from $\{0,1\}^*$, we can map it to element in $(\mathbb{Z}_p^*)^k$ using a collision resistent cryptographic hash function $H_1 : \{0,1\}^* \to (\mathbb{Z}_p^*)^k$. Also, we assume that the messages to be encrypted are elements in $\mathbb{G}_1$.

The HIBE scheme [15] consists of the following algorithms:

$\mathsf{Setup}(\ell) \rightarrow (\mathsf{para}, \mathsf{msk})$: To generate system parameters for an HIBE of maximum depth $\ell$, select a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p^*$, and set $g_1 = g^\alpha$. Next, pick random elements $g_2, g_3, h_1, \cdots, h_\ell \in \mathbb{G}$. The public parameters and the master secret key $\mathsf{msk}$ are

$$\mathsf{para} = (g, g_1, g_2, g_3, h_1, \cdots, h_\ell), \qquad \mathsf{msk} = g_2^\alpha.$$

$\mathsf{KeyGen}(\mathsf{para}, d_{\mathsf{ID}_{k-1}}, \mathsf{ID}_k) \rightarrow d_{\mathsf{ID}_k}$: Let $d_{\mathsf{ID}_{k-1}} = (g_2^\alpha \cdot (h_1^{I_1} \cdots h_{k-1}^{I_{k-1}} \cdot g_3)^{r'}, g^{r'}, h_k^{r'}, \cdots, h_\ell^{r'}) = (a_0, a_1, b_k, \cdots, b_\ell)$. To generate a private key $d_{\mathsf{ID}_k}$ for an identity $\mathsf{ID}_k = (I_1, \cdots, I_k) \in (\mathbb{Z}_p^*)^k$ of depth $k \leq \ell$, pick a random $t$ from $\mathbb{Z}_p^*$, and output

$$d_{\mathsf{ID}_k} = \Big(a_0 \cdot b_k^{I_k} \cdot (h_1^{I_1} \cdots h_k^{I_k} \cdot g_3)^t, a_1 \cdot g^t, b_{k+1} \cdot h_{k+1}^t, \cdots, b_\ell \cdot h_\ell^t\Big).$$

$\mathsf{Enc}(\mathsf{para}, \mathsf{ID}, M) \rightarrow \mathsf{CT}$: To encrypt a message $M$ under the public key $\mathsf{ID} = (I_1, \cdots, I_k) \in (\mathbb{Z}_p^*)^k$, pick a random $s \in \mathbb{Z}_p^*$, and output $\mathsf{CT} = (e(g_1, g_2)^s \cdot M, g^s, (h_1^{I_1} \cdot h_k^{I_k} \cdot g_3)^s)$.

$\mathsf{Dec}(\mathsf{para}, d_{\mathsf{ID}}, \mathsf{CT}) \rightarrow M$: Given a ciphertext $\mathsf{CT} = (A, B, C)$ under identity $\mathsf{ID} = (I_1, \cdots, I_k)$, using the private key $d_{\mathsf{ID}} = (a_0, a_1, b_{k+1}, \cdots, b_\ell)$, the algorithm outputs $M \leftarrow A \cdot \frac{e(a_1, C)}{e(B, a_0)}$.

*2) Modification on the above HIBE:* In order to design a path authentication protocol, we modify the Setup and Enc algorithms, and add three more algorithms to HIBE, which are $\mathsf{SetSecretKey}, \mathsf{SetPublicKey}$ and $\mathsf{ReEnc}$. Our tag path authentication protocol within an EPCIS consists of the following seven algorithms:

$\mathsf{Setup}(\ell) \rightarrow (\mathsf{para}, \mathsf{msk})$: Each issuer $I$, which is the root of an HIBE tree, is responsible for the system's setup. Besides the output of all the system's public parameters and the master secret key listed in HIBE, we require that the issuer chooses a collision resistent cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$, and a reversible but not homomorphic mapping $\mathcal{F}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$. This algorithm also generates a private key $sk_{\mathsf{epcis}} = H(\mathsf{ID}_{\mathsf{epcis}})^\alpha$ for its EPCIS. At last, $sk_{\mathsf{epcis}}$ is sent to the EPCIS via a secure channel.

After the generation of the system's parameters, the issuer publishes para on EPCDS, and keeps $\mathsf{msk} = \alpha$ as secret.

$\mathsf{KeyGen}(\mathsf{para}, d_{\mathsf{ID}_{k-1}}, \mathsf{ID}_{R_k}) \rightarrow d_{\mathsf{ID}_k}$: It is identical to that of HIBE.

$\mathsf{Enc}(\mathsf{para}, \mathsf{msk}, \mathsf{ID}_{R_i}, \mathsf{ID}_T) \rightarrow C_{\mathsf{ID}_{R_i}}$: Since a tag's identity $\mathsf{ID}_T$ is an arbitrary string from $\{0,1\}^*$, we need $\mathcal{F}_1$ to map it into an element of $\mathbb{G}_1$. The encryption in our solution is

$$\mathsf{CT} = (e(g_1, g_2)^s \cdot \mathcal{F}_1(\mathsf{ID}_T), g^s, (h_1^{I_1} \cdot g_3)^s),$$

where $I_1$ is the identity of the chosen reader $R_i$ by issuer. We require that the issuer sign on the $\mathsf{ID}_T$ in addition to the ciphertext CT. The issuer's signature on $\mathsf{ID}_T$ is $\sigma_1 = (\sigma_{1_1}, \sigma_{1_2}) = (H(\mathsf{ID}_T \| I_0), (H(\mathsf{ID}_T \| I_0))^\alpha)$, where $I_0$ is the identity of the issuer. At last, $C_{\mathsf{ID}_{R_i}} = (\mathsf{CT}, \sigma_1)$ is the output which is written to the tag as its initial state.

$\mathsf{SetSecretKey}(\mathsf{para}, R_i) \rightarrow x_{R_i}$: It selects a random $x_{R_i} \in \mathbb{Z}_p^*$ as reader $R_i$'s secret value.

$\mathsf{SetPublicKey}(\mathsf{para}, x_{R_i}) \rightarrow \mathsf{PK}_{R_i}$: On input para and $x_{R_i}$, it returns public key $\mathsf{PK}_{R_i} = g^{x_{R_i}}$.

$\mathsf{ReEnc}(\mathsf{para}, x_{R_i}, C_{\mathsf{ID}_{R_i}}, \mathsf{ID}_R) \rightarrow C_{\mathsf{ID}_R}$: This algorithm transforms a state from a parent reader to the one under its son reader's public key. When reader $R_i$ reads out the state of a tag in depth $k$ under it, where $R_i$'s identity is represented as $\mathsf{ID}_k = (I_1, \cdots, I_k), (1 \leq k < \ell)$, it computes the state $C_{k+1} = (c_{k+1_1}, c_{k+1_2}, c_{k+1_3})$ in depth $k+1$ under reader $I_{k+1}$ as follows (a simple way to choose the next reader is the neighbor of $R_i$):

- Denote the identifier in depth $k+1$ to be $(I_1, \cdots, I_{k+1})$;
- Parse the ciphertext $C_{\mathsf{ID}_{R_i}}$ in depth $k$ as $(c_{k_1}, c_{k_2}, c_{k_3}, \sigma_k = (\sigma_{k_1}, \sigma_{k_2}))$;
- The reader checks whether the following equation holds: for $k = 1$, $e(g_1, \sigma_{k_1}) \stackrel{?}{=} e(g, \sigma_{k_2})$; for $k > 1$, $e(\mathsf{PK}_{R_i}, \sigma_{k_1}) \stackrel{?}{=} e(g, \sigma_{k_2})$, where $\mathsf{PK}_{R_i}$ is the public key of reader $R_i$.
- Select a random element $s_k \in \mathbb{Z}_p^*$, and compute

$$\begin{aligned} c_{k+1_1} =& c_{k_1} \cdot e(g_1, g_2)^{s_k} \cdot e(g^{r_{k+1}}, h_{k+1}^{I_{k+1}})^{s+s_k} \\ & \cdot \prod_{i=2}^{k} (e(c_{k_2}, h_i^{I_i})^{r_{k+1}} \cdot e(g^{r_{k+1}}, h_i^{I_i})^{s_k}), \\ c_{k+1_2} =& c_{k_2} \cdot g^{s_k}, c_{k+1_3} = c_{k_3} \cdot (h_1^{I_1} \cdot g_3)^{s_k}, \end{aligned}$$

  where $r_{k+1}$ is the random element which the parent reader selected for generating the private key of its son readers (corresponding to the random number $t$ in the KeyGen algorithm in [15]). We assume that each reader maintains such a private table for all of its son readers. When a parent reader chooses a new reader as its son in a supply chain, it stores this secret value in the table. Meanwhile, a reader generates private key for this son reader using the secret value. The secret value is also used to re-encrypt a ciphertext each time the parent reader transforms a ciphertext to its son. It is reasonable to make the above assumption, since both random elements and a son's private key are generated by its parent in an HIBE scheme. Note that in HIBE, the private key of a son reader is generated by its parent using the parent's private key and a random number, i.e., $r_{k+1}$. Thus it is impossible for an entity to generate a private key for a node if it does not know the private key of node's parent.
- $R_i$ signs $\sigma_k$. We denote by $\sigma_{k+1}$ the signature. If $k = 1$, $\sigma_{k+1} = (\sigma_{k+1_1}, \sigma_{k+1_2}) = (H(e(g_1, \sigma_{k_1}) \| R_i), H(e(g_1, \sigma_{k_1}) \| R_i)^{x_{R_i}})$. Otherwise, if $k > 1$, $\sigma_{k+1} = (\sigma_{k+1_1}, \sigma_{k+1_2}) = (H(e(\mathsf{PK}_{R_j}, \sigma_{k_1}) \| R_i), H(e(\mathsf{PK}_{R_j}, \sigma_{k_1}) \| R_i)^{x_{R_i}})$, where $\mathsf{PK}_{R_j}$ is the public key of reader $R_i$'s parent.
- $C_{\mathsf{ID}_R} = (C_{k+1}, \sigma_{k+1})$ is the output which is written to the tag as its new state under reader $R$.

$\mathsf{Dec}(\mathsf{para}, d_{\mathsf{ID}_k}, C_{\mathsf{ID}_{R_k}}) \rightarrow (T, \perp)$: Identical to that of HIBE.

Finally, the leaf node, i.e., the last reader $R_k$ in a valid path, decrypts the tag's identity, and checks the validation of the tag and its path. If the check succeeds, the leaf node encrypts the

tag's identity and its path with its EPCIS's public key using identity-based encryption scheme, and signs the ciphertext. The encryption is like: $C' = (C'_1, C'_2, C'_3, \sigma_k)$, where

$$(C'_1, C'_2, C'_3) = \big(g^s, \mathcal{F}_1(\mathsf{ID}_T \| P_T) \cdot e(g_1, H(\mathsf{ID}_{\mathsf{epcis}})^s), H(C'_1 \| C'_2)^s\big),$$
$$\sigma_k = \big(H(e(\mathsf{PK}_{R_j}, \sigma_{k-1_1}) \| R_k), H(e(\mathsf{PK}_{R_j}, \sigma_{k-1_1}) \| R_k)^{x_{R_k}}\big).$$

Here, $P_T$ is the path which tag $T$ with identity $\mathsf{ID}_T$ goes through, $\mathsf{PK}_{R_j}$ is the public key of reader $R_k$'s parent reader $R_j$, and $(\sigma_{k-1_1}, \sigma_{k-1_2})$ is the signature output by $R_j$.

On receiving the ciphertext $C' = (C'_1, C'_2, C'_3, \sigma_k)$, the EPCIS first verifies the ciphertext by checking if the following equations hold: $e(C'_1, H(C'_1 \| C'_2)) \stackrel{?}{=} e(g, C'_3); e(\mathsf{PK}_{R_k}, \sigma_{k_1}) \stackrel{?}{=} e(g, \sigma_{k_2})$. If both of the equations hold, the EPCIS computes $\mathcal{F}_1(\mathsf{ID}_T \| P_T) = C'_2 / e(C'_1, sk_{\mathsf{epcis}})$. As $\mathcal{F}_1$ is a reversible function, we can compute tag's identity and its path from $\mathcal{F}_1(\mathsf{ID}_T \| P_T)$.

Further, the EPCIS verifies the path by checking the signature $\sigma_k$. For example, if a valid path is $I_0 \to R_1 \to \cdots \to R_k$. The EPCIS checks whether the following equation holds:

$$e\Big(\mathsf{PK}_{R_k}, H\Big(e\Big(\cdots, H\Big(e\big(\mathsf{PK}_{R_1}, H(e(g_1, H(\mathsf{ID}_T \| I_0)) \| R_1)\big) \cdots\Big) \| R_k\Big)\Big) \stackrel{?}{=} e(g, \sigma_{k_2}).$$

*3) Implementation of Path Authentication:* In this section, we use a concrete example to show how the tag path authentication is performed within an EPCIS. In Fig. 3, there are five readers in an EPCIS. Suppose that three tags $T_1$, $T_2$, and $T_3$ go through the EPCIS via three different paths as follows:

1) $T_1$: $I \to R_1 \to R_2 \to R_5$
2) $T_2$: $I \to R_1 \to R_3$
3) $T_3$: $I \to R_4 \to R_5 \to R_1$

In Fig. 3, $R_5$ appears two times, in depth two and three, respectively. Suppose that the depth of the tree is three, the private keys of $R_5$ in depth two and three are $(g_2^{\alpha} \cdot (h_1^{I_4} h_2^{I_5} g_3)^{r'_5}$, $g^{r'_5}$, $h_3^{r'_5})$ and $(g_2^{\alpha} \cdot (h_1^{I_1} h_2^{I_2} h_3^{I_5} g_3)^{r''_5}$, $g^{r''_5})$, respectively.

$R_5$ may interact with more than one tags, by checking the signature which is a part of the state of tag $T_1$, $R_5$ can determine whether the state is re-encrypted from $R_2$ or $R_4$. Thus $R_5$ knows the right private key for decrypting the right state. If the signature is valid for $R_2$, $R_5$ verifies the path for $\mathsf{ID}_{T_1}$ in two steps:

(1) With the private key generated by its parent reader $R_2$, $R_5$ checks if the following equation holds: $e(g, g_2^{\alpha} \cdot (h_1^{I_1} h_2^{I_2} h_3^{I_5} g_3)^{r''_5}) \stackrel{?}{=} e(g_1, g_2) \cdot e(g^{r''_5}, (h_1^{I_1} h_2^{I_2} h_3^{I_5})) \cdot e(g^{r''_5}, g_3)$.

(2)$R_5$ verifies the path for $\mathsf{ID}_{T_1}$ further as follows:

- Obtain $\mathsf{ID}_{T_1}$ by decrypting the ciphertext with the private key generated by $R_2$.
- Check if the following equation holds:

$$e\Big(\mathsf{PK}_{R_2}, H\Big(e\Big(\mathsf{PK}_{R_1}, H\big(e\big(g_1, H(\mathsf{ID}_{T_1} \| I)\big) \| R_1\big)\Big) \| R_2\Big)\Big) \stackrel{?}{=} e(g, \sigma_{2_2}).$$

Note that $R_5$ is a leaf node now. If the above verifications are successful, it can decrypt the ciphertext from the tag, and then re-encrypt it with its EPCIS's public key, and signs on the encrypted message.

### *B. Path Authentication among* EPCIS

When a product is moving to another EPCIS for further disposal, the current EPCIS needs to verify the tag's identity and the path within it as described in Section III-A2. Then, we adopt the batch verification signature [16] for path authentication among EPCIS. In such process, the previous EPCIS could choose to sign a batch of messages which are ready to be sent to the same EPCIS. This saves the verification cost since the next EPCIS can batch verify the signature upon different products and different paths using two pairing operations only. When the next EPCIS's verification on these tags and their paths are accepted, it decrypts the tag and its path and executes our path authentication protocol to proceed these tags further within the new EPCIS.

Camenisch et al. [16] proposed several methods on batch verifier for messages from (possibly) different signers on (possibly) different messages with or without random oracles. The verification time of their algorithms is independent of the number of signatures. In this paper, we adopt their batch verification of BLS signature [17]. Some slight modifications are made to make it suitable in our situation. We denote by $(H(\mathsf{ID}_{\mathsf{epcis}})^{\mathsf{ssk}}, \mathsf{ssk})$ the verification/signing key pair of an EPCIS, where $H$ is a collision-resistant cryptographic hash function. Meanwhile, instead of $n$ users signing $n$ different messages, we consider one user signing $n$ different messages. The revised algorithms are given below:

$\mathsf{Gen}(\mathsf{para})$: The EPCIS selects a random element $sk_{\mathsf{epcis}} \in \mathbb{Z}_p^*$, and sets $\mathsf{svk}_{\mathsf{epcis}} = H(\mathsf{ID}_{\mathsf{epcis}})^{sk_{\mathsf{epcis}}}$, $\mathsf{ssk}_{\mathsf{epcis}} = sk_{\mathsf{epcis}}$. The $\mathsf{svk}_{\mathsf{epcis}}$ is published on EPCDS. This can be performed when the issuer setups the systems parameters. Here the para is the same as those in the tag path authentication protocol.

$\mathsf{Sign}(\mathsf{para}, \mathsf{ID}_T, P_T, \mathsf{ssk}_{\mathsf{epcis}_i}, pk_{\mathsf{epcis}_j})$: If $\mathsf{EPCIS}_i$ intends to sign the combination of a tag and its path, it encrypts the combination and signs the message $\mathsf{Enc}_{pk_{\mathsf{epcis}_j}}(\mathcal{F}_1(\mathsf{ID}_T \| P_T))$ as $\sigma_{\mathsf{epcis}_i} = H(\mathsf{Enc}_{pk_{\mathsf{epcis}_j}}(\mathcal{F}_1(\mathsf{ID}_T \| P_T)))^{\mathsf{ssk}_{\mathsf{epcis}_i}}$. Since $\mathcal{F}_1$ maps an element from $\{0,1\}^*$ to $\mathbb{G}_1$, we require that $\mathsf{Enc}_{pk_{\mathsf{epcis}_j}}(H(\mathsf{ID}_T \| P_T))$ be an arbitrary asymmetric cryptographic encryption implemented in bilinear group $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ and the message space is $\mathbb{G}_1$.

Batch Verify: Given $n$ purported signatures $\sigma_i$ on $n$ different messages $m_i$ under the same $\mathsf{EPCIS}_j$ for $i = 1, \cdots, n$, the EPCIS equipped with the qualified private key, $\mathsf{EPCIS}_k$, first checks whether or not $\sigma_i \in \mathbb{G}$ for all $i$ and outputs reject if not. Otherwise, $\mathsf{EPCIS}_k$ computes $h_i = H(m_i)$ and generates a vector $\delta = (\delta_1, \cdots, \delta_n)$, where each $\delta_i$ is a random element from $\mathbb{Z}_p^*$, and $m_i$ is a ciphertext on a specific tag and its path. $\mathsf{EPCIS}_k$ checks on the equation $e(\Pi_{i=1}^n \sigma_i^{\delta_i}, g) \stackrel{?}{=} \Pi_{i=1}^n e(h_i, pk_{\mathsf{epcis}_j})^{\delta_i}$. If this equation holds, it outputs accept; otherwise, it outputs reject.

If the verification succeeds, $\mathsf{EPCIS}_k$ decrypts the ciphertext using its private key. The tag's identity and its path in the previous EPCIS are disclosed. Now a new round of path

authentication starts within the new EPCIS. Here, we keep the tag and its path private by encrypting them using the next EPCIS's public key.

## IV. Security Analysis

We only provide brief security analysis for space reason, while detailed proofs will be given in a full version.

**Theorem IV.1.** *Any forged state of a tag $T_i$ output by a* PPT *adversary $\mathcal{A}$, which $\mathcal{A}$ has claimed that the tag $T_i$ has gone through some step but in fact the tag does not go through it in a supply chain, can be detected. Thus, our protocol has the authentication property.*

Suppose that a tag $T_c$'s state is output by $\mathcal{A}$, and $\mathcal{A}$ has claimed that $T_c$ has gone through reader $R^*$, but in fact $T_c$ has not gone through it in a valid path $P$. $\mathcal{C}$ can detect it easily. $\mathcal{C}$ first checks the signature part of the state to get $R$'s parent. Next, using the private key generated by $R$'s parent, $\mathcal{C}$ decrypts the state. If $\mathcal{C}$ gets $T_c$, it gets the right path. Third, $\mathcal{C}$ verifies the path using $R$'s private key. If the verification succeeds, and $R^*$ appears in the path, it means that $T_c$ appears in a valid path containing $R^*$. $\mathcal{C}$ further verifies the signature $\sigma_i$ using all the reader identities in the valid path. If the signature is acceptable, it means that $T_c$ must have gone through reader $R^*$ and $R^* \in P$. In this case, $\mathcal{C}$ outputs 1; Otherwise, it means that $T_c$ has not gone through reader $R^*$, but $R^* \in P$. In this case, $\mathcal{C}$ outputs 0, where $P$ is the true path $T_c$ has gone through. In both cases, $\mathcal{C}$ can detect the true status.

**Theorem IV.2.** *If the HIBE [15] scheme is secure, then, our path authentication protocol has the tag privacy.*

Note that the path is verified in two steps. First, using its private key, the reader can verify the path. Second, when getting the identity of tag $T_c$, the reader could further verify the path by checking if the signature is a valid one according to the sequence of the readers appearing in the path. From this point of view, $\mathcal{A}$ does not have any knowledge of the path if he does not know the private key of the reader (neither can the adversary verify the path nor decrypt the tag and its path without the private key, which is ensured by the security of HIBE scheme.). Thus $\mathcal{A}$ cannot determine if $T_c$ has passed through the target reader $R^*$.

**Theorem IV.3.** *If there exists a* PPT *adversary $\mathcal{A}$ that can break the tag unlinkability of our protocol, then there exists another* PPT *adversary $\mathcal{B}$ who can break the CCA-security of the HIBE scheme [15].*

IND-CCA security implies tag unlinkability directly. In both cases, the adversary is required to distinguish a ciphertext correspond to which of two given messages.

**Theorem IV.4.** *If there exists a* PPT *adversary $\mathcal{A}$ who can break the path unlinkability of our protocol, then there exists a* PPT *adversary $\mathcal{B}$ who can break the CCA-security of the HIBE scheme [15].*

If there is a PPT adversary who can tell if these two tags go through an identical path, we can construct another adversary that distinguishes these two tags from some states read out from some reader in the path, that breaks the IND-CCA security of HIBE. Tag unlinkability implies path unlinkability. However, the reverse does not hold.

## V. Conclusion

In this paper, we present a two-level path authentication protocol in an RFID-based supply chain in EPCglobal Network with desirable properties, including dynamic path generation and distributed authentication. Our solution is a significant improvement over the previous work [3], which deals with static and determined path and centralized authentication. Our solution is more suitable for large-scale dynamic supply chain management as demanded in practice nowadays.

## Acknowledgment

The work of the second author is supported in part by the Office of Research at Singapore Management University. The work of other authors is supported in part by the NSFC under grant No 61033014, and NSFC A3 Foresight Program under grant No. 61161140320.

## References

[1] T. Mao, J. Williams, and A. Sanchez, "Interoperable internet scale security framework for RFID networks," in *ICDE Workshops*, 2008, pp. 94–99.

[2] R. H. Deng, Y. Li, M. Yung, and Y. Zhao, "A new framework for rfid privacy," in *ESORICS*, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds. Springer, 2010, pp. 1–18.

[3] E.-O. Blass, K. Elkhiyaoui, and R. Molva, "Tracker: Security and privacy for RFID-based supply chains," in *NDSS*, 2011, pp. 455–472.

[4] T. Staake, F. Thiesse, and E. Fleisch, "Extending the epc network: the potential of RFID in anti-counterfeiting," in *SAC*, 2005, pp. 1607–1612.

[5] B. King and X. Zhang, "Securing the pharmaceutical supply chain using RFID," in *MUE*. IEEE Computer Society, 2007, pp. 23–28.

[6] E. P. SToP, "Stop tampering of products," 2010, http://www.stop-project.eu.

[7] Motorola, "Saudi arabia's luxury retailer jade jewellery implements motorola's RFID technology to improve inventory management and security. http://tinyurl.com/yg6wzjv," 2010.

[8] TAGSYS RFID, "RFID luxury goods solutions," 2010, http://www.tagsysrfid.com/markets/industries/luxury-goods.

[9] S. E. Sarma, S. A. Weis, and D. W. Engels, "RFID systems and security and privacy implications," in *CHES*, 2002, pp. 454–469.

[10] S. Vaudenay, "On privacy models for RFID," in *ASIACRYPT*, ser. LNCS, K. Kurosawa, Ed., vol. 4833. Springer, 2007, pp. 68–87.

[11] Y. Li and X. Ding, "Protecting RFID communications in supply chains," in *ASIACCS*, F. Bao and S. Miller, Eds. ACM, 2007, pp. 234–241.

[12] A. Juels and S. A. Weis, "Defining strong privacy for RFID," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, 2009.

[13] C. Ma, Y. Li, R. H. Deng, and T. Li, "RFID privacy: relation between two notions, minimal condition, and efficient construction," in *CCS*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 54–65.

[14] C. Berbain, O. Billet, J. Etrog, and H. Gilbert, "An efficient forward private RFID protocol," in *CCS*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 43–53.

[15] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *EUROCRYPT*, ser. LNCS, R. Cramer, Ed., vol. 3494. Springer, 2005, pp. 440–456.

[16] J. Camenisch, S. Hohenberger, and M. O. Pedersen, "Batch verification of short signatures," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, M. Naor, Ed., vol. 4515. Springer, 2007, pp. 246–263.

[17] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *J. Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.